Published in final edited form as:

Methods Mol Biol. 2015 ; 1260: 165–178. doi:10.1007/978-1-4939-2239-0_10.

GENN: A GEneral Neural Network forLearning Tabulated Data With Examples From Protein Structure Prediction

Eshel Faraggi,

Department of Biochemistry and Molecular Biology, Indiana University School of Medicine, Indianapolis, Indiana 46202, USA; Battelle Center for Mathematical Medicine, Nationwide Children's Hospital, Columbus, Ohio 43215, USA; and Physics Division, Research and Information Systems, LLC, Carmel, Indiana, 46032, USA, phone: 317-332-0368

Andrzej Kloczkowski

Andrzej Kloczkowski Battelle Center for Mathematical Medicine, Nationwide Children's Hospital, Columbus, Ohio 43215, USA; and Department of Pediatrics, The Ohio State University, Columbus, Ohio 43215, USA

Eshel Faraggi: efaraggi@gmail.com; Andrzej Kloczkowski: Andrzej.Kloczkowski@nationwidechildrens.org

Abstract

We present a GEneral Neural Network (GENN) for learning trends from existing data and making predictions of unknown information. The main novelty of GENN is in its generality, simplicity of use, and its specific handling of windowed input/output. Its main strength is its efficient handling of the input data, enabling learning from large datasets. GENN is built on a two layered neural network and has the option to use separate inputs-output pairs or window based data using data structures to efficiently represent input-output pairs. The program was tested on predicting the accessible surface area of globular proteins, scoring proteins according to similarity to native, predicting protein disorder and has performed remarkably well. In this paper we describe the program and its use. Specifically, we give as an example the construction of a similarity to native protein scoring function that was constructed using GENN. The source code and Linux executables for GENN are available from Research and Information Systems at http://mamiris.com and from the Battelle Center for Mathematical Medicine at http://mathmed.org. Bugs and problems with the GENN program should be reported to EF.

Keywords

Neural Network; Protein Scoring; Windowed Input; Automatic Learning; GENN; Protein Structure Prediction

1 Introduction

Proteins perform their function through structure based spatial and temporal interactions and give rise to the biosphere as we know it. More knowledge about them will contribute to our fundamental understanding of life and in practical terms will revolutionize medicine, bioscience, and other fields. However, in the context of present day computational and analytical tools they are too large and too complex. This is best exemplified in the ever

growing gap between the number of known protein structures (relatively few) and the number of known protein sequences (relatively many). As of the end of 2013, there are under 100,000 structures deposited in the Protein Data Bank but over 33 million protein sequences from about 30,000 different organisms in the NCBI RefSeq database. In many other areas of human endeavor constantly growing amount of information is being collected. Frequently, this information can be useful in other, related or unrelated, human endeavors (1–15). Consumer purchasing history is an example where the past purchases of a consumer group can be used to assign a probability distribution of future purchases, and could, for example, give advertisers critical information for marketing strategy. In protein structure prediction too, experimentally solved protein structures can be used to infer the structure of unsolved proteins.

These vast amounts of data require specialized tools to extract useful information from them. Here we present one such tool that can be used to analyze large amounts of information and learn from it based on examples. The learning is achieved through steepest descent training of a two layer neural network. It is able to efficiently handle instance-based, and window-based training as will be described below. Its novelty is in its ability to handle any quantitative information, limited only by hardware, and that it can efficiently store window-based data, enabling modification of the modeling approaches without the need to modify the database. For large window-based training these are unique features as far as we could find. We term this tool GENN for GEneral Neural Network.

GENN was programmed in FORTRAN 90. In its design and implementation its efficiency and usability were of major concern. It is constructed out of several subroutines that process the data, initialize the model, and train it. It is also capable of producing predictions from existing single models or producing ensemble predictions with expected deviations. Its execution is terminal based. It was built on Ubuntu Linux, under the BASH (Bourne Again SHell) environment. For the rest we will use that environment to describe usage.

Although GENN can handle any data, it was designed and built for questions related to the relationship between the residue sequence of a protein and its native structure. Therefore we shall use terminology from that field to describe various features of the program. In general terms the problem is as follows. We are given the residue sequence of a protein, which can be derived from the genetic code. This residue sequence is in essence the chemical formula of the protein molecule. In turn we are to provide the best prediction for the physical structure of this molecule. However, since this is a difficult problem, certain approximations are usually made, either by coarse graining the full atom model, or by looking at derived structural properties such as the accessible surface area (16–26).

The general architecture of GENN is given in Fig. 1. The user supplied information includes the feature files which include information that will be used to establish a prediction, and the training files which contain the information that is to be predicted. The error between the neural network predicted value and the training values is used to train the weights using the steepest descent back propagation method (21). These weights are between the inputs through hidden layers one and two (H1 and H2 respectively in Fig. 1) and to the prediction output.

2 Types of training and prediction in GENN

2.1 Instance-Based Prediction

We use the term *instance-based prediction* to describe cases in which each example to be learned from is independent of every other example in the database. This is in contrast to *windows-based prediction*, to be discussed next, where examples covered in overlapping windows will share common features. All programs related to instance-based prediction have a base name of ‘genn2inst’.

An example of instance-based prediction is prediction of global protein features. When one wants to predict quantities for entire protein sequences, for example the radius of gyration, one is dealing with instance-based prediction. For each protein various input features are gathered but in general different proteins will have unique input features that are not shared by other proteins. Another example we have implemented using GENN, that will be described below, is that of predicting the fitness of model protein structures to the native state. Such a scheme was implemented successfully using GENN for the CASP10 experiment (27).

2.2 Window-Based Prediction

Window-based prediction is used to describe cases where different instances, with different outputs, share common inputs. Maybe the easiest case to describe is that of predictions for properties of residues along a protein sequence. In this case, we use sequentially neighboring residues (within a window of a certain width) to obtain the input features, then as we go along the chain neighboring residues will share common input features. In this case it is wasteful to record the inputs for each instance (residue) separately. It is more efficient to only store the input/output features of the protein and extract the appropriate inputs and outputs appropriate for each residue. All programs related to window-based prediction have a base name of ‘genn2wind’. More discussion of window-based prediction is given in earlier works (21, 25, 28–31).

Boundary conditions are important for window-based prediction. For example for residues near a protein terminus the sliding window may fall out of range. These considerations are internally controlled in GENN. Windows that encompass regions outside the boundary of the instance are truncated to include only those parts that fall within range.

Because of the relative complexity associated with partitioning a database of windowed instances we designed the genn2wind version to automatically perform an n-fold cross validation, where n is determined by the command line options. By default n is equal to ten and the last partition is taken as the test set. In this mode 5% out of the remaining training data is used for over-fit protection. If one wishes to use only an over-fit protection set without a test set, for example while developing a server, this can be achieved using the ‘-sv’ option. This built-in n-fold cross validation mode is limited to the window-based programs. For the instance-based programs only an over-fit set is chosen by default.

3 Database and Initialization File

In general the database is composed of separate directories each representing a given instance, a protein in our example. Each directory contains the input and output files associated with each instance. In our example the output file contains a list of the normalized ASA values to be predicted for each residue. The exact details of this normalization will be given later in the text when RASA will be defined. The input files then should have quantities that have predictive power with respect to the desired output. Specific details of the input and output files will be given when discussing ASAquick.

The initialization file is used to describe the locations of the required information for a given run. An example is provided in Fig. 2. The first line in the file gives the path of the head directory where all the instance directories are located. This should be enclosed in double quotes to preserve the slashes in the path name. The next line describes the inputs used, which are assumed the same for all instances. Each input is represented by a file name that will contain its values for a particular instance in its instance directory. Multiple inputs should be separated by spaces. In a similar way the third line lists the outputs to be predicted, represented by files in the instance directory that contain these values. Multiple outputs should be separated by spaces. The design of GENN is intended to facilitate changing both inputs and outputs for testing and research. The remaining lines list the instances on which the training will be performed. For the case of instance-based prediction this lists individual instances. For window-based prediction this lists a collection of instances grouped together in single files, for example the residues in a given protein. To allow for user control GENN does not modify the order of the list of instances. Randomization of the order of training instances, which can be useful, is left to the user.

One should note that for window-based prediction, since each input file can contain many instances (residues) an additional requirement on the database is imposed. To ascertain a match between inputs and outputs for a given instance two identifiers are used (originally the crystallographer index and the residue type for proteins). During the reading of the database both identifiers are checked and the program halts on mismatches between different files.

4 Training and Prediction

The specifics of using GENN for training and predicting from data are given below.

4.1 Training A Model

The program is run from the command line. The options associated with training are given in Table 1. By default the program looks for a file called ‘genn.in’ to read the training dataset structure from (initialization file). If this file does not exist in the directory it is necessary to include in the command line a ‘-l’ option followed by the name of the initialization file.

Given additional data a model can be retrained by including it as the initial condition for the new round of training. This is achieved using the ‘-wf’ option followed by the name of the

file containing the weights and parameters of the trained model. In this case the weights and other model parameters are used as initial values and no randomization is carried out. Then new training is performed on the new instances.

4.2 Predictions From Trained Models

Once a model has been trained all parameters are stored in a file. This file then can be used to give a prediction for new instances. In addition a collection of models can be used to produce an average prediction with an accompanying standard deviation. This standard deviation can prove beneficial in cases where quality of prediction is desired, but this requires testing on specific cases. The basic options associated with prediction from a trained model are given in Table 1. Note that some options are common between training and prediction tasks, however, model parameters such as the activation parameter or others that are related to the model architecture are stored in, and read from, the model file and should be reissued only in the uncommon event that one desires to override their values.

5 Special Options

Several unique features are provided with GENN. These options grew out of several applications related to protein structure prediction (21, 25, 29–31) In this section we outline their possible uses. Refer to Table 1 for the required syntax. These options are available for the window-based programs.

5.1 Filter Network

Some cases for window-based prediction can benefit from a filter, where predictions over a window are used as inputs for a follow up prediction. Examples include protein secondary structure where a filter prediction layer is often used (21, 25, 29). While a filter layer can always be run using a second-stage neural network, for the window-based programs a filter layer can be included by using an option in the command line. Refer to Table 1 for the appropriate syntax. By default the filter network runs with 11 nodes, this value can be changed as needed. Note that a specific name is used to save the filter model. When getting predictions from trained filter models the filter option should also be used.

5.2 Guided Learning

In some instances of window-based prediction the relevance of a given location to a particular prediction site can be dependent on the distance between the location and the prediction site. In these cases it may be helpful to guide the network in that direction. One way of achieving this is by introducing an extra set of weights that have such distance dependence (21). This is the default behavior for the window-based program. An option is available to switch off this behavior.

5.3 Global Inputs and Outputs

For certain window-based prediction global variables may be important. We shall consider the case of protein secondary structure to illustrate the point. In a window-based approach the secondary structure of a given residue is considered in the context of a window around it. However, certain parameters such as the amino acid composition of the protein may contain

information about general tendencies to form a specific protein class (all-alpha, all-beta, alpha/beta or alpha + beta). This information would typically not be contained in a limited window view and hence can help the prediction. Indeed global input features can significantly contribute to the prediction accuracy. To use global inputs/outputs, one should store their values in files called genn.gin/genn.gut respectively in the directory corresponding to that instance. An option, '-gi' for global input and/or '-go' for global output, is required in the command line to include these global files. These options should be followed by the number of values to use from these global files enabling control of how much global information to use. Note that options for global files are only required for training. The input structure is recorded in the weight files and is retrieved from them when called to predict.

5.4 Degeneracy Training

In some cases of window-based prediction one may like to sample certain cases more often than others. For example, for disordered proteins one may like to sample more of the disordered residues in a given protein instance (31). In instance-based prediction one can include instances and their occurrence at will by repeating them in the file 'genn.in'. For window-based prediction to change the sampling frequency within an instance, one includes a file with an integer count of the number of times (including zero) to train on a given training output. Hence, the degeneracy file order should match the rest of the input/output and should include the corresponding indexes. Degeneracy files should be stored in instance directories under files named 'genn.dgn', and an option should be given for their use.

5.5 Types of Output

GENN is able to predict multiple outputs. This in turn enables specific functionality besides general multi-output prediction. The first option is to treat the multiple outputs as components of a multi-state probability vector and train, optimize, and report accordingly. While training is not affected by this option, optimization is carried on the ability of the trained network to correctly distinguish the most dominant component of the probability vector. Reporting also follows the same protocol. This behavior was found useful when an ancestor of GENN was used to predict protein secondary structure (25).

Another possibility is to predict several instances of the same output and generate an average prediction from the same trained network. This has proven beneficial in several instances of continuous value prediction (29). It provides an extra layer of averaging, with more control over variation between prediction models and hence the ability to average over specific random errors in certain circumstances. This also generates an estimate for the prediction stability in the reported standard deviation over predictions and in this form may be beneficial in assigning prediction stability and accuracy.

6 Automated Learning

GENN was started as the first, and simplest, component of an automated learning machine. Automated learning here means being able to "sit" on a big database (e.g., the Internet) and answer in meaningful ways questions posed by either humans or machines. A sort of Watson

[wikipedia.org/wiki/Watson_(computer)] but without human design. Rather, continuously, progressively and automatically developing an image of the database pertinent to the questions it was exposed to in its existence. Here we specifically mean a non-memorizing learner, it is expected that on certain questions, different learners will respond differently depending on their histories. In this respect GENN was set up to create local networks on the fly.

The other, more complex component of such a learning machine would require to extract information in a meaningful way to describe both input and output features of a general question/answer pair. Note that some questions have multiple acceptable answers. Also, it would require building a self-sustaining learning mechanism. Both these tasks are monumental and would require breaking down the various problems further.

7 Examples

In this section we describe work that grew out of this version of GENN. Older versions of GENN helped produce an NMR fluctuation predictor (30) and a protein disorder predictor termed SPINE-D (31), as well as some other applications. GENN was also involved in various testing in several labs. SPINE-D participated in the CASP9 experiment and was ranked among the top five methods (32). This version of GENN was also used to create ASAquick, a fast accessible surface area predictor that uses only single sequence information (manuscript in preparation), and a knowledge based protein scoring function termed Seder (27) which we will present here as an example of use of GENN. Seder participated in the CASP10 (32) experiment as the “Kloczkowski_Lab” group. It was ranked second in predicting the structure of the ‘hard’ targets category (33), third for ‘all’ targets. It was the only prediction approach that was ranked in the top three for both ‘hard’ and ‘all’ cases.

This is a good place to point out that for any practical problem, the machine learner is only secondary to the problem of representation and interpretation of the input and the output. Care and thought of how to present the input and extract the output can produce great improvements in learning and prediction quality. To a second degree, given the same input/output information, different quality learners can produce different outcomes. For example, it has been our general experience that smaller sets of data are better learnt by methods such as support vector machines while larger sets are better learnt by neural networks, the difference being several relative percent accuracy. On the other hand, for example, changes of the representation of the input/output for predicting dihedral angles of proteins has resulted in almost 100% reduction in the prediction error (18, 21, 28, 29).

Seder (27) is an example of an implementation of GENN. It was developed specifically for structured proteins, to rank structural models according to their similarity to a native structure. In the directory ‘example/Seder’ of the source code distribution we give examples for the feature files used to make Seder. We use the same names to introduce them here, more information is available in the Seder paper (27). The first line of the input file ‘genn.in’ gives the directory location of the database of instances. In the second line of ‘genn.in’ the inputs are listed. typ233w.norm contains information about the distance of individual atoms

to the solvent. There are a total of 936 real numbers making up this feature. res2res refers to the partial energy sums between pairs of residues. There are a total of 441 real numbers making up this feature. 4bod, dfire2, and rwplus refer to the four-body (34–36), DFIRE2 (37, 38), and RWPlus (39) potentials respectively. The desired output from the neural network, given on the third line, is a transformation of the TM-score (40, 41) used to measure similarity between native and model structures (decoys). For training we used server models from the Critical Assessment of protein Structure Prediction (CASP) (32) rounds 5 through 9 (94717 structures). The fourth line of 'genn.in' points to the single example of an instance given here. This points to subdirectories of the 'db' directory. Actual training will involve a list of such instances. We average over several training realizations with different initial conditions. For each, in addition to randomizing the initial weights, we also randomize the list of training proteins (PDB+CASP) and select thirty thousand of them. From this subset, 30% were used to compose the over-fit-protection set while the remaining 70% was used for on-line training. Over-fit testing was done after each training epoch.

The command line for training one realization of a Seder neural network is given by 'nohup time ./genn2inst.e -l seder.in -f 0.3 -mi 200 -h1 51 -h2 31 -r $RANDOM &'. We have included three features of Linux/BASH that are not directly related to this work but are nontheless useful. The first is the 'nohup' command which allow a training of the weights even after logout or loss of connection. Note that you must have the ampersand symbol at the end of the line for this functionality. The second is the 'time' command which is useful in estimating run times and optimization. The third is '$RANDOM' which is a BASH reserved word for generation of pseudorandom numbers, it comes in handy in automation of neural network creation since seeding is done independently of your application and depends on the system state (hence arbitrary). The training command line given above will train a set of weights and output them with other necessary information for prediction into a file. By design the weights file name is constructed uniquely from the process ID and other parameters associated with the run. Here we shall assume its name to be 'wei.out'. A single prediction from this file for a single protein with ID, PID, would look something like './genn2inst.e -wf wei.out -pr1 PID'. Where is it assumed that the directory mentioned in the beginning of wei.out contains a subdirectory called PID with the necessary files. Note that GENN is designed to calculate prediction error, hence it will look for target files even if asked to predict. If such a target file does not exist, is not in your possession, or a complete blind test of the software is sought, trivial (zero filled) target files should be generated. Of course then the reported prediction errors are meaningless. For a collection of proteins with PIDs listed in the file 'list.PID' and taking the average over a collection of weight files listed in file 'list.wei' prediction and prediction variation are obtained via the command './genn2inst.e -aw list.wei -prl list.PID'. Both these prediction methods produces a text output to screen which can be piped to a text file. If a list of proteins is given the first column in the output is the PID listed in 'list.wei'. The program 'awk' can then be easily used to generate individual prediction files if those are necessary. Note that we have given an example of instance-based prediction, the syntax for window-based prediction is identical. Example files for window-based prediction are in the subdirectory 'db/pdb.window'.

8 Summary

We have presented GENN, a general neural network designed to train on ad-hoc data. GENN was designed with efficiency and modularity in mind as part of a more complex algorithm of an automated learner. It can take any numerical input/output problem and prepare a corresponding, non-memorizing, model structure to represent this data. Data can be organized in files containing individual instances or a collection of ordered instances where each line is an individual input/output target. GENN is available from Research and Information Systems at http://mamiris.com, and from the Battelle Center for Mathematical Medicine at http://mathmed.org.

Acknowledgments

We gratefully acknowledge the financial support provided by the National Institutes of Health (NIH) through Grants R01GM072014 and R01GM073095 and the National Science Foundation through Grant NSF MCB 1071785. Both authors would like to thank the organizers of CASP10 conference in Gaeta, Italy, for inviting them to the conference and providing free registration to EF. EF would also like to thank Keith Dunker for hosting him at IUPUI.

References

1. Kassin SM. 1979; Consensus information, prediction, and causal attribution: A review of the literature and issues. Journal of Personality and Social Psychology. 37:1966.
2. Crick NR, Dodge KA. 1994; A review and reformulation of social information-processing mechanisms in children's social adjustment. Psychological Bulletin. 115:74.
3. Fielding AH, Bell JF. 1997; A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation. 24:38–49.
4. Makhoul J. 1975; Linear prediction: A tutorial review. Proceedings of the IEEE. 63:561–580.
5. Fontenot RJ, Wilson EJ. 1997; Relational exchange: a review of selected models for a prediction matrix of relationship activities. Journal of Business Research. 39:5–12.
6. Rost B, et al. 2001; Review: protein secondary structure prediction continues to rise. Journal of Structural Biology. 134:204–218. [PubMed: 11551180]
7. Maier HR, Dandy GC. 2000; Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environmental Modelling & Software. 15:101–124.
8. Chang JC, Wooten EC, Tsimelzon A, Hilsenbeck SG, Gutierrez MC, Elledge R, Mohsin S, Osborne CK, Chamness GC, Allred DC, et al. 2003; Gene expression profiling for the prediction of therapeutic response to docetaxel in patients with breast cancer. The Lancet. 362:362–369.
9. Schofield W, et al. 1985; Predicting basal metabolic rate, new standards and review of previous work. Human Nutrition. Clinical Nutrition. 39:5. [PubMed: 4044297]
10. Blundell T, Sibanda B, Sternberg M, Thornton J. 1987; Knowledge-based prediction of protein structures. Nature. 326:26.
11. Chou PY, Fasman GD. 1978; Empirical predictions of protein conformation. Annual Review of Biochemistry. 47:251–276.
12. Floudas C, Fung H, McAllister S, Mönnigmann M, Rajgaria R. 2006; Advances in protein structure prediction and de novo protein design: A review. Chemical Engineering Science. 61:966–988.
13. Moult J. 2005; A decade of casp: progress, bottlenecks and prognosis in protein structure prediction. Current Opinion in Structural Biology. 15:285–289. [PubMed: 15939584]
14. Vazquez A, Flammini A, Maritan A, Vespignani A. 2003; Global protein function prediction from protein-protein interaction networks. Nature Biotechnology. 21:697–700.

15. Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel HP. 2005; Protein function prediction via graph kernels. Bioinformatics. 21:i47–i56. [PubMed: 15961493]
16. Chothia C. 1974; Hydrophobic bonding and accessible surface area in proteins. Nature. 248:338–339. [PubMed: 4819639]
17. Moret M, Zebende G. 2007; Amino acid hydrophobicity and accessible surface area. Physical Review E. 75:011920.
18. Dor O, Zhou Y. 2007; Real-spine: An integrated system of neural networks for real-value prediction of protein structural properties. PROTEINS: Structure, Function, and Bioinformatics. 68:76–81.
19. Durham E, Dorr B, Woetzel N, Staritzbichler R, Meiler J. 2009; Solvent accessible surface area approximations for rapid and accurate protein structure prediction. Journal of molecular modeling. 15:1093–1108. [PubMed: 19234730]
20. Zhang H, Zhang T, Chen K, Shen S, Ruan J, Kurgan L. 2009; On the relation between residue flexibility and local solvent accessibility in proteins. Proteins: Structure, Function, and Bioinformatics. 76:617–636.
21. Faraggi E, Xue B, Zhou Y. 2009; Improving the prediction accuracy of residue solvent accessibility and real-value backbone torsion angles of proteins by guided-learning through a two-layer neural network. Proteins: Structure, Function, and Bioinformatics. 74:847–856.
22. Zhang T, Zhang H, Chen K, Ruan J, Shen S, Kurgan L. 2010; Analysis and prediction of rna-binding residues using sequence, evolutionary conservation, and predicted secondary structure and solvent accessibility. Current Protein and Peptide Science. 11:609–628. [PubMed: 20887256]
23. Gao J, Zhang T, Zhang H, Shen S, Ruan J, Kurgan L. 2010; Accurate prediction of protein folding rates from sequence and sequence-derived residue flexibility and solvent accessibility. Proteins: Structure, Function, and Bioinformatics. 78:2114–2130.
24. Nunez S, Venhorst J, Kruse CG. 2010; Assessment of a novel scoring method based on solvent accessible surface area descriptors. Journal of chemical information and modeling. 50:480–486. [PubMed: 20356089]
25. Faraggi E, Zhang T, Yang Y, Kurgan L, Zhou Y. 2012; Spine x: Improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. Journal of Computational Chemistry. 33:259–267. [PubMed: 22045506]
26. Wang C, Xi L, Li S, Liu H, Yao X. 2012; A sequence-based computational model for the prediction of the solvent accessible surface area for α-helix and β-barrel transmembrane residues. Journal of Computational Chemistry. 33:11–17. [PubMed: 21935968]
27. Faraggi E, Kloczkowski A. 2013; A global machine learning based scoring function for protein structure prediction. Proteins: Structure, Function, and Bioinformatics. doi: 10.1002/prot.24454
28. Xue B, Dor O, Faraggi E, Zhou Y. 2008; Real value prediction of backbone torsion angles. Proteins: Structure, Function, and Bioinformatics. 72:427–433.
29. Faraggi E, Yang Y, Zhang S, Zhou Y. 2009; Predicting continuous local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. Structure. 17:1515–1527. [PubMed: 19913486]
30. Zhang T, Faraggi E, Zhou Y. 2010; Fluctuations of backbone torsion angles obtained from nmr-determined structures and their prediction. Proteins: Structure, Function, and Bioinformatics. 78:3353–3362.
31. Zhang T, Faraggi E, Xue B, Dunker AK, Uversky VN, Zhou Y. 2012; Spine-d: accurate prediction of short and long disordered regions by a single neural-network based method. Journal of Biomolecular Structure and Dynamics. 29:799–813. [PubMed: 22208280]
32. Moult J, Fidelis K, Kryshtafovych A, Tramontano A. 2011; Critical assessment of methods of protein structure prediction (casp) round ix. Proteins: Structure, Function, and Bioinformatics. 79:1–5.
33. CASP10. [accessed 10-June-2012] Official group performance ranking. 2012. Online; http://www.predictioncenter.org/casp10/groups_analysis.cgi

34. Feng Y, Kloczkowski A, Jernigan R. 2007; Four-body contact potentials derived from two protein datasets to discriminate native structures from decoys. Proteins: Structure, Function, and Bioinformatics. 68:57–66.

35. Feng Y, Kloczkowski A, Jernigan RL. 2010; Potentials' r'us web-server for protein energy estimations with coarse-grained knowledge-based potentials. BMC bioinformatics. 11:92. [PubMed: 20163737]

36. Gniewek P, Leelananda SP, Kolinski A, Jernigan RL, Kloczkowski A. 2011; Multibody coarse-grained potentials for native structure recognition and quality assessment of protein models. Proteins: Structure, Function, and Bioinformatics. 79:1923–1929.

37. Zhou H, Zhou Y. 2002; Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. Protein Science. 11:2714–2726. [PubMed: 12381853]

38. Yang Y, Zhou Y. 2008; Ab initio folding of terminal segments with secondary structures reveals the fine difference between two closely related all-atom statistical energy functions. Protein Science. 17:1212–1219. [PubMed: 18469178]

39. Zhang J, Zhang Y. 2010; A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction. PloS one. 5:e15386. [PubMed: 21060880]

40. Zhang Y, Skolnick J. 2004; Scoring function for automated assessment of protein structure template quality. Proteins: Structure, Function, and Bioinformatics. 57:702–710.

41. Xu J, Zhang Y. 2010; How significant is a protein structure similarity with tm-score= 0.5? Bioinformatics. 26:889–895. [PubMed: 20164152]

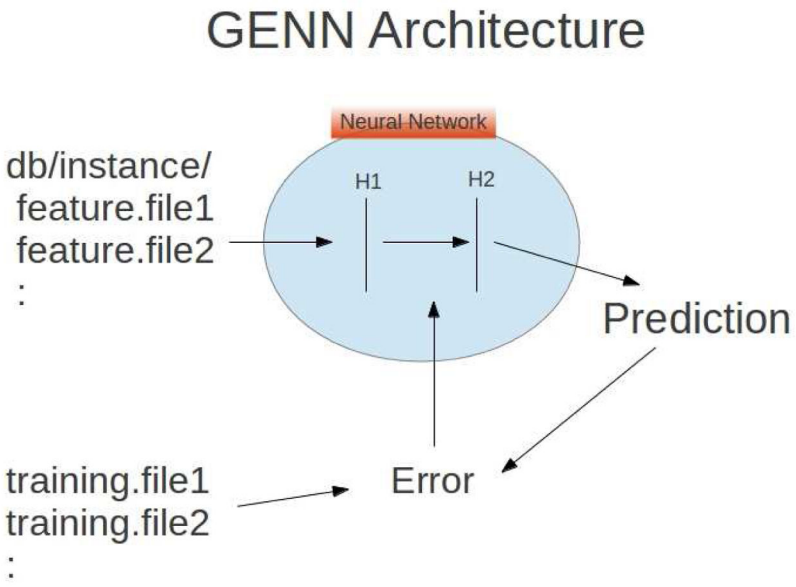

Fig. 1.

Schematic diagram describing the architecture of GENN. The feature files include information used to establish a prediction. The training files contain information to be predicted. The error between the neural network predicted value and the training values is used to train its weights. H1 and H2 refer to the first and second hidden layers respectively.

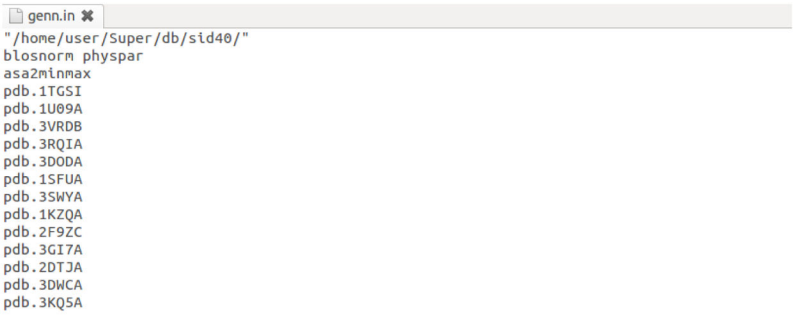

Fig. 2.

Example of the input file for GENN. The first line gives the head directory where all the instance directories are. This should be enclosed in double quotes to preserve the slashes. The next line describes the inputs used separated by spaces. The third line lists the outputs to be predicted represented by file names separated by spaces. The remaining lines list the instances on which the training will be performed.

Table 1

GENN Options: Options available for GENN. Some options either do not have a default but require a value or are logical and do not require a value.

Category[a]	Flag[b]	Description	Default[c]
Train	-l	db list file	genn.in
	-owf	Weights output file	
	-m	Maximum number of epochs	1000
	-np	Number of database files to use	whole db
	-r	Random seed	time dependent
	-cv	Test fold index	10
	-f	Fraction of database for test fold	0.1
	-wi	Input window size	21
	-wo	Output window size	1
	-h1	Number of hidden layer one nodes	21
	-h2	Number of hidden layer two nodes	21
	-mi	Maximum number of bad iterations	100
	-a	Activation parameter	0.2
	-u	Learning rate	0.001
	-p	Momentum	0.4
	-hf	Number of filter hidden layer nodes	11
	-nf	Run network filter	
	-gf	Guiding factor	2.d0
	-ng	Don’t use distance guiding	
	-gi	Number of global inputs	0
	-go	Number of global outputs	0
	-df	Use degeneracy files	
Predict	-pr1	Predict single file (give id)	
	-prl	File of ids to predict	
	-dw	Not same weight types, reread database	
	-aw	Average over weights in file	
Both	-d	Database head directory	
	-wf	Weights input file	
	-so	Average over outputs (nvo)	
	-po	Probability output	
	-h	Print help	

Options available for GENN.

[a] We distinguish three types of categories for options. ‘Train’ for training specific options. ‘Predict’ for prediction specific options. And, ‘universal’ for options that used in both.

[b] The flag or command line option.

[c] Default values, some flags either do not have a default but require a value or are logical and do not require a value.